

Lösungen

Hier findest du die Lösungen zu den Fragen und Aufgaben. Wo es Quellcode gibt, ist nur der abgedruckt der die Beantwortung der Fragen betrifft. Es kann also sein, dass du die Anweisung erst in eine Prozedur einbinden musst, damit du sie sinnvoll nutzen kannst.

Die Entwicklungsumgebung

1. Um die Entwicklungsumgebung aufzurufen, startest du Excel und drückst `[Alt] + [F11]`.
2. Werden Makros nicht ausgeführt, liegt das in aller Regel am eingestellten Makrovirenschutz? Den kannst du über das Menü Extras in Excel einstellen.
3. Der VBA-Code wird in Excel immer in einer Excel-Arbeitsmappe gespeichert. Um Code zu öffnen, musst du also die Arbeitsmappe öffnen, in der sich der Code befindet.
4. Der Projektextplorer ist ein Fenster in der Entwicklungsumgebung in dem du alle VBA-Projektdateien also die Module und UserForms verwalten kannst.
5. Wenn du Anweisungen im Direktfenster ausführen willst, gibst du die Anweisung dort ein und schließt sie mit `[Enter]` ab.
6. Die VBA-Hilfe wird standardmäßig nicht installiert. Sie wird nur bei einer Vollinstallation installiert oder wenn du bei der benutzerdefinierten Installation die Hilfe explizit mit installierst.
7. Wenn du die Hilfe zu einer VBA-Anweisung aufrufen willst, setzt den Cursor in die Anweisung und drücke `[F1]`.

Code strukturieren

1. Variablen können ihren Wert innerhalb des Programmablaufs ändern, Konstanten behalten immer ihren Wert.
2. Eine Prozedur ist benannte Abfolge von Anweisungen, die über ihren Namen aufgerufen werden kann.
3. Unterprozeduren können keine Wert zurückgeben. Dazu benötigst du eine Funktion.

4. Eine Prozedur kann beliebig viele Parameter haben.
5. Wenn du eine Prozedur aufrufen willst, nennst du deren Namen und gibst danach die Parameter in der gewünschten Reihenfolge an: `Prozedurname Wert1, Wert2, ...`. Benötigst du bei Funktionen den Rückgabewert, fasst du die Parameter in Klammern ein.
6. Parameter die optional sein sollen, definierst du, indem du dem Parameter das Schlüsselwort `Optional` voranstellst.
7. Prozeduren mit dem Namen `Auto_Open` werden automatisch ausgeführt, wenn die Arbeitsmappe geöffnet wird.

Einfache Berechnungen

1. Ein Ausdruck ist eine Anweisung, die nichts tut, sondern dafür einen Wert hat. Sie besteht aus mindestens einem Wert und einem Operator. Beide zusammen stellen den Wert des Ausdrucks dar.
2. Der Operator `/` führt eine normale Division durch, der Operator `\` führt hingegen eine ganzzahlige Division, und `Mod` liefert den Rest, der bei einer ganzzahligen Division verbleibt.
3. Der Ausdruck `(True AND False)` hat den Wert `False`, weil nur ein Operand `True` ist.
4. Klammern in komplexeren Ausdrücken dienen dazu festzulegen, welche Teile eines Ausdrucks zuerst berechnet werden.
5. Wenn du prüfen möchtest, ob ein bestimmtes Zeichen oder eine Zeichenfolge in einer Zeichenkette vorkommt, verwendest du dazu die `Instr`-Funktion.
6. Der Ausdruck muss lauten: `Mid(Text, 1, 1) = "*" "`.

Verzweigungen und Schleifen

1. Eine Abweisende Schleife prüft eine Eintrittsbedingung, eine nicht abweisende eine Austrittsbedingung. Wenn die Eintrittsbedingung einer abweisenden Schleife nicht erfüllt ist, wenn sie das erste Mal betreten werden soll, wird die Schleife gar nicht erst betreten. Eine nicht abweisende Schleife wird immer mindestens einmal ausgeführt.
2. Die Do-Loop-Until-Schleife ist eine nicht abweisende Schleife, da die Schleifenbedingung erst im Schleifenfuß steht.
3. Um eine Schleife zu erstellen, die von 100-50 herunter zählt und dabei für jeden zweiten Wert ausgeführt wird, musst du die Schleife wie folgt formulieren:

```
For I=100 To 50 Step -2
```

```
...
```

```
Next I
```

4. Zulässig ist eine negative Step-Angabe schon, auch wenn der Anfangswert kleiner als der Endwert einer For-Schleife ist. Allerdings wird dann die For-Schleife nicht betreten. In VBA 5.0-Hostanwendungen entsteht dann jedoch eine Endlosschleife.
5. Wenn du in einer Verzweigung mehr als eine Bedingung überprüfen willst, brauchst du eine Mehrfachverzweigung, bspw. If-Then-ElseIf-Else oder musst mehrere If-Verzweigungen ineinander verschachteln.
6. Wenn in einer If-Then-ElseIf-Else-Anweisung mehr als ein Ausdruck wahr ist, wird nur der Then-Zweig der ersten erfüllten Bedingung ausgeführt und dann die Schleife verlassen.

Zugreifen auf Excel Daten und Zellen

1. Beide Eigenschaften, `ThisWorkbook` und `ActiveWorkbook`, geben ein `Workbook`-Objekt zurück. `ActiveWorkbook` gibt jedoch die aktive Arbeitsmappe zurück, während `ThisWorkbook` die Arbeitsmappe zurückgibt, die den Code enthält. Beide können also die gleiche Arbeitsmappe zurückgeben, das muss aber nicht zwingend der Fall sein.

2. Die `Sheets`-Auflistung enthält alle Blätter der Arbeitsmappe, die `Worksheets`-Auflistung nur die Tabellenblätter?
3. Eine Schleife, die alle Zellen der ersten Spalte eines Tabellenblattes durchläuft könnte z. B. so aussehen:


```
Dim Zelle As Range
For Each Zelle In _
ActiveWorkbook. _
Worksheets(1).Columns(1).Cells
...
Next Zelle
```
4. Die `Value`-Eigenschaft eines `Range`-Objekts gibt den Wert einer Zelle zurück.
5. Der Unterschied zwischen den Eigenschaften `Formula` und `FormulaLocal` besteht darin, dass die `FormulaLocal`-Eigenschaft die Formel in der Sprache der Excel-Version enthält. In deutschen Excel-Versionen werden also die deutschen Funktionsnamen verwendet. Die `Formula11`-Eigenschaft speichert die Formel immer in Englisch.
6. Mit dem `Interior`-Objekt kannst du die Formatierungen des Zellinneren, wie Muster und Füllfarbe festlegen.
7. Arbeitsmappen werden mit der `Open`-Methode der `Workbooks`-Auflistung geöffnet und mit `Close` geschlossen.
8. Der Aufruf `RGB(256,0,0)` ist nicht gültig, da für die einzelnen Farbwerte nur Werte von 0 bis 255 einschließlich zulässig sind.
9. Bei Zuweisung eines Farbewertes an die `Color`-Eigenschaft kannst du jede Farbe verwenden, die im RGB-Farbraum zulässig ist. Nutzt du die `ColorIndex`-Eigenschaft, kannst du nur die Farben der Excel-Farbpalette verwenden.

Excel: Eigenschaften manipulieren und ermitteln

1. Die Excel-Version kannst du mit der Version-Eigenschaft des Application-Objekts ermitteln.
2. Es ist in der Regel nicht möglich, über die Version-Eigenschaft des VBE-Objekts die VBA-Version zu ermitteln, weil durch den Makrovirenschutz ab Office 2002 der Zugriff auf das Objekt VBE unterbunden wird.
3. Eine Anweisung, die das Standardarbeitsverzeichnis von Excel auf "C:\Daten\" setzt muss wie folgt lauten:
Application.DefaultFilePath="C:\Daten\".
4. Das AutoStart-Verzeichnis von Excel kannst du mit der StartupPath-Eigenschaft ermitteln. Mit Debug.Print Application.StartupPath wird der AutoStart-Ordner im Testfenster ausgegeben.
5. Eine solche Prozedur muss auf jeden Fall eine Funktion sein und könnte wie folgt aussehen:

```
Function Geoeffnet(AMName As _
    String) As Boolean
    Dim AM As Workbook
    For Each AM In _
        Application.Workbooks
        If UCase(AM.Name) = _
            UCase(AMName) Then
            Geoeffnet = True
            Exit Function
        End If
    Next AM
End Function
```

Benutzeroberflächen gestalten

1. Mit der InputBox-Funktion kannst du einen Eingabedialog auf dem Bildschirm anzeigen lassen und damit Werte vom Benutzer einlesen lassen.
2. Meldungen kannst du mit der MsgBox-Funktion ausgeben.
3. Der Aufruf der MsgBox-Funktion muss folgendermaßen aussehen, damit ein JA- und ein NEIN-Button sowie ein Fragezeichen als Symbol angezeigt werden:

```
MsgBox "Meldung", vbYesNo + _
    vbQuestion
```

4. UserForms sind spezielle Klassenmodule mit denen du eine Benutzeroberfläche gestalten kannst. Du kannst Steuerelemente wie Eingabefelder, Auswahllisten und Schaltflächen einfügen, mit denen der Benutzer dann die Anwendung bedienen kann.
5. UserForms werden mit der Show-Methode der UserForm angezeigt. Heißt die UserForm z.B. UserForm1 lautet die Anweisung dazu UserForm1.Show.
6. Zunächst musst du die Userform ausblenden und dann mit Unload entladen. Der Code dazu lautet:

```
Me.Hide
Unload Me
```

7. Gibt es eine Möglichkeit einen Dialog zur Dateiauswahl anzeigen zu lassen?
8. Schlüsselwort Me im Code einer UserForm stellt das Objekt dar, das zur Laufzeit aus dem Klassenmodul erzeugt wurde. Darüber können Sie alle Methoden und Eigenschaften der UserForm abrufen.

Auf Ereignisse und Fehler reagieren

1. Ein Ereignis ist ein vordefinierter Zustand, bei dem der normale Programmablauf unterbrochen wird.
2. Um eine Ereignisprozedur für das `Click`-Ereignis einer Schaltfläche in einer UserForm zu erstellen klickst du doppelt auf die Schaltfläche. Die IDE erzeugt dann automatisch die Ereignisprozedur.
3. Das Ereignis `Initialize` einer UserForm tritt ein, wenn die UserForm geladen wird.
4. Eine Ereignisprozedur für das `Initialize`-Ereignis, die die Hintergrundfarbe der UserForm ändert, könnte so aussehen:

```
Private Sub UserForm_Initialize()  
    Me.BackColor = _  
        RGB(255, 255, 255)  
End Sub
```
5. Haltepunkte dienen dazu, stellen im Code zu markieren, bei denen in den Debug-Modus geschaltet werden soll.
6. Willst du dafür sorgen, dass Laufzeitfehler nicht zur Unterbrechung des Programmablaufs führen fügst du vor der Anweisung, die einen Fehler auslösen könnte, die Anweisung `On Error Resume Next` ein.
7. Die `Resume`-Anweisung in der Fehlerbehandlung bewirkt, dass die Anweisung wiederholt wird, die den Fehler ausgelöst hat.
8. Die Beschreibung des letzten Fehlers kannst du über die `Description`-Eigenschaft des `Err`-Objekts auslesen. Die Anweisung muss also lauten:
`Debug.Print Err.Description.`